(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開平5－28002
(43)公開日　平成5年(1993)2月5日

(51)Int.Cl.⁵	識別記号	庁内整理番号	Ｆ１	技術表示箇所
Ｇ０６Ｆ　11/28	３２０　Ｅ	8725－5Ｂ		
	３１０　Ｆ	8725－5Ｂ		

審査請求　未請求　請求項の数4(全 8 頁)

(21)出願番号　特願平3－184244

(22)出願日　平成3年(1991)7月24日

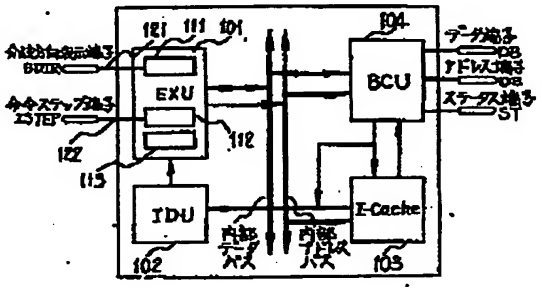
(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目7番1号

(72)発明者　新井　智久
東京都港区芝五丁目7番1号日本電気株式会社内

(74)代理人　弁理士　内原　晋

(54)【発明の名称】　マイクロプロセッサ

(57)【要約】
【目的】内蔵命令キャッシュを使用した状態で、命令実行状況のリアルタイム・トレースをサポートするマイクロプロセッサにより、キャッシュ・メモリを内蔵したマイクロプロセッサのプログラムおよびシステムのデバッグを容易にする。
【構成】分岐命令の分岐方向を検出して表示する分岐方向検出手段１１１、命令処理の実行開始を検出して命令実行開始検出手段１１２、分岐先を静的に計算できないイベントの発生を検出する動的分岐検出手段１１３を持つ。



1

【特許請求の範囲】
【請求項1】　キャッシュ・メモリを内蔵したマイクロプロセッサにおいて、命令の実行開始を外部に通知する手段と、分岐命令の分岐方向を外部に通知する手段と、プログラムの流れをあらかじめ静的に計算できない場合に割り込みを発生する手段とを有することを特徴とするマイクロプロセッサ。
【請求項2】　前記命令の実行開始を外部に通知する手段として、専用の端子を使用することを特徴とする請求項1記載のマイクロプロセッサ。
【請求項3】　前記命令の実行開始を外部に通知する手段として、専用の端子上に出力するパルス信号を利用することを特徴とする請求項1記載のマイクロプロセッサ。
【請求項4】　前記分岐命令の分岐方向を外部に通知する手段として、専用の端子を使用することを特徴とする請求項1記載のマイクロプロセッサ。
【発明の詳細な説明】
【0001】
【産業上の利用分野】本発明は、命令コードあるいはメモリ・オペランドをキャッシングするキャッシュ・メモリを内蔵したマイクロプロセッサに関し、特にキャッシュ・メモリを内蔵したマイクロプロセッサのシステム開発あるいはプログラム開発におけるデバッグに関する。
【0002】
【従来の技術】マイクロプロセッサのプログラム開発時、あるいはシステム開発時におけるデバッグの方法としては大別して以下のようなものがある。
1．トレース：プログラムの実行位置を示す情報、あるいはメモリ・オペランドのアクセスに関する情報を収集し、これを組み立てることにより命令実行あるいはオペランド・アクセスの順序を知る。前記情報は、一般にマイクロプロセッサの外部端子で観測可能なバス・サイクルのアドレス、データ、およびステータス等から構成することができる。
【0003】一般に上記情報は、マイクロプロセッサの外部で収集することができるので、マイクロプロセッサに特別な機能がなくてもトレース機能は実現できる。また、デバッグの対象となるプログラムの実行に対して、割込み／例外を発生させて中断したり、バス・サイクルを待合せしたりする必要がないので、デバッグ状態／非デバッグ状態でタイミングが変わらないという利点がある。
【0004】一方、デバッグ対象のプログラムに対して、特定の状況を検出して中断しないため、プログラムの状態を単に観測するだけの受動的な方法である。
2．トラップ：あらかじめ設定した特定の位置（アドレス）の命令コードあるいはメモリ・オペランドのアクセスがあったことでデバッグ・プログラム（以下デバッガと称する）に制御を移し、さらに詳細なデバッグを進める。

2

【0005】トレースとは異なり、必要な時点でデバッガに制御を移し、デバッグを進めることができる利点を持つ。ただし、設定したアドレスに対し正確に（例えば設定したアドレスを持つ命令コードを実行した直後に／実行する直前に）割込み／例外を発生し、デバッガに制御を移すためのハードウェア的な機構（一般にトラップ割込みと呼ばれる）を持つことが要求される。
【0006】また、1）プログラムの正確な実行順序を知っておく必要がある、2）予定どおりのトラップがかけられなければ（例えばトラップよりも前にプログラムが暴走するような場合）プログラム実行を中断できない、などの欠点を持つ。
3．シングル・ステップ：1命令を実行する毎にデバッガに制御を移し、マイクロプロセッサの内部状態（汎用レジスタ、プロセッサ・ステータス・ワード：ＰＳＷ、プログラム・カウンタ：ＰＣなど）を表示したり、一部分の内容を変更しながら、プログラムの実行を進める。
【0007】マイクロプロセッサの内部状態を命令実行毎に把握できるため、極めて詳細にプログラムの実行経過を把握することができる。一方、マイクロプロセッサ自体に1命令を実行した時点で割込み／例外を発生し、デバッガに制御を移すためのハードウェア的な機構（一般にシングル・ステップ割込みと呼ばれる）を持つことが要求され、どのマイクロプロセッサでも実現可能なわけではない。また、1命令毎にデバッガに制御が移るため、デバッグ対象のプログラムに対し、1）実行効率が悪い、2）タイミング・クリチカルな処理（例としてタイマ・ルーチンなどが挙げられる）はデバッグできない、3）内部動作タイミングがデバッグ／非デバッグ時で全く異なる、等の欠点もある。
【0008】以上に述べたように各デバッグ方法には、それぞれの利点と欠点があるため、一般には次のように組み合わせて使用する。
【0009】1．トレース機能により実際にどのような順序で命令が実行されているかを知る。
【0010】2．トラップ機能により問題が発生している付近でデバッガに制御を移す。
【0011】3．シングル・ステップ機能により1命令づつ丹念に命令の実行状況を追う。
【0012】このように、トレース機能は効率的なプログラムおよびシステムのデバッグを行う上で必須の機能である。
【0013】以下では、トレース機能の実現方法について図面を参照して説明する。
【0014】図6は、従来のキャッシュ・メモリを内蔵したマイクロプロセッサの構成を示している。また図7は、トレース機能を実現するための一般的なシステム構成を示している。
【0015】はじめに図6，7を参照して内蔵キャッシ

ュを使用しない場合のマイクロプロセッサの動作、およびトレース・システムの動作について簡単に説明する。内蔵キャッシュを使用しない場合の動作については後述する。

【００１６】はじめに図６を用いて、内蔵キャッシュを使用しない場合のマイクロプロセッサの内部動作について説明する。実行ユニット６０１は、次に実行する命令のアドレスを生成して内部アドレスを介してバスサイクル制御ユニット６０４に供給する。バスサイクル制御ユニット６０４は、実行するバスサイクルが命令フェッチ・サイクルであることを示すステータス信号をステータス端子ＳＴに、フェッチする命令のアドレスをアドレス端子ＡＢにそれぞれ出力し、データ端子ＤＢに返された命令コードを読み込んで命令デコード６０２に転送する。命令デコード６０２は与えられた命令コードをデコードし、制御信号を生成して実行ユニット６０１に供給する。実行ユニット６０１は与えられた制御信号に従ってそれぞれの命令に応じた処理を実行する。

【００１７】それでは、上述したマイクロプロセッサの命令実行をトレースする場合のトレース動作について図７を参照して簡単に説明する。

【００１８】マイクロプロセッサ７０１は、内蔵命令キャッシュを使用しないため、命令をフェッチする場合には常に命令フェッチ・サイクルを起動して、命令コードをメモリより読みだしてくる必要がある。命令フェッチ要求が生じると、マイクロプロセッサ７０１は命令フェッチ・サイクルを起動して、ステータス信号およびフェッチする命令のアドレスをそれぞれ出力する。この命令フェッチ・サイクルに応答してメモリ７０２が返す命令コードを、データバスを介して内部に読み込んで、命令コードに対応する命令を実行する。ところで、トレース・システム上のトレース・アナライラ７０３は、ステータス信号を監視することによりこの命令フェッチ・サイクルの発生を検出し、ストローブ信号７１１を発生させて命令フェッチ・サイクル中にアドレスバス上に出力されている命令アドレスを、トレース・メモリ７０４に取り込んで記録する。このようにして、マイクロプロセッサ７０１が実行する命令のアドレスをすべてトレース・メモリ７０４に記録することにより命令トレースを作成する。

【００１９】

【発明が解決しようとする課題】さて、図６に戻って内蔵命令キャッシュを使用する場合のマイクロプロセッサの動作について考えてみる。

【００２０】まず、実行ユニット６０１は、次に実行する命令のアドレスを生成して内部アドレスバスを介して命令キャッシュ６０３に供給する。

【００２１】命令キャッシュ６０３に該当する命令が登録されている場合（以下「ヒット」と言う）、命令キャッシュ６０３は直ちに命令コードを命令デコーダ６０２に供給する。命令デコーダ６０２に命令コードが供給されて以降の動作は、前述した内蔵命令キャッシュを使用しない場合の動作と同様である。

【００２２】一方、該当する命令が命令キャッシュ６０３に登録されていない場合（以下「ミスヒット」と言う）、命令キャッシュ６０３はバスサイクル制御ユニット６０４に、ミスヒットした命令のメモリからのフェッチと、命令キャッシュ６０３への登録を要求する。バスサイクル制御ユニット６０４はリプレース・サイクルを起動して、アドレスおよびステータスを各端子に出力し、ミスヒットした命令コードをデータ端子から読み込んで命令キャッシュ６０３に登録するとともに命令デコーダ６０２に供給する。命令デコーダ６０２に命令コードが供給されて以降の動作は、前述した内蔵命令キャッシュを使用しない場合の動作と同様である。

【００２３】したがって、マイクロプロセッサ内部の命令処理が進行しても、命令キャッシュがヒットしている場合には、命令処理の進行を示す情報が外部に出力されないため、外部信号を観測することによってマイクロプロセッサの命令処理の進行状況を知ることはできない。したがって、図７に示すような構成のトレース・システムでは、マイクロプロセッサの動作をトレースすることは不可能となる。

【００２４】この問題を解決するために、トレース時には命令キャッシュの機能を不活性にして、すべての命令フェッチを外部から観測可能にして、トレース機能を実現する方法が考えられる。しかしながら、この場合トレース時と実動作時のマイクロプロセッサの動作が異なってしまうため、正確なデバッグができなくなるという欠点がある。

【００２５】

【課題を解決するための手段】本発明は、キャッシュ・メモリを内蔵したマイクロプロセッサにおいて、命令の実行開始を外部に通知する手段と、分岐命令の分岐方向を外部に通知する手段と、プログラムの流れをあらかじめ静的に計算できない場合に割り込みを発生する手段とを有している。

【００２６】

【実施例】以下、図面により詳述する。

【００２７】図１は本発明を利用したマイクロプロセッサの構成を示している。図中では簡単のため本発明に関連するブロックのみ示し、それ以外は省略してある。

【００２８】１０１は命令処理の実行ユニット、１０２は命令デコーダ、１０３は命令キャッシュ、１０４はバス制御ユニットである。これらのユニットは、内部アドレスバスおよび内部データバスを介して接続されている。実行ユニット１０１は、その内部に分岐命令の分岐方向を検出する分岐方向検出手段１１１、各命令の実行開始を検出する命令実行開始検出手段１１２、プログラムの流れがあらかじめ静的に計算できないイベント（た

とえば、レジスタ間接分岐命令の実行、例外の発生による例外処理プログラムへの分岐等）の発生を検出する手段１１３（以下「動的分岐検出手段」と呼ぶ）を内蔵している。

【００２９】まず、本マイクロプロセッサの動作を説明する。

【００３０】実行ユニット１０１は、次に実行すべき命令を決定しそのアドレスを内部アドレスを介して命令キャッシュ１０３に与える。命令キャッシュ１０３がヒットした場合、命令キャッシュ１０３は直ちに命令コードを命令デコーダ１０２に供給する。命令デコーダ１０２は命令コードをデコードして制御信号を生成し、実行ユニット１０１に供給する。実行ユニット１０１は命令デコーダ１０２から供給された制御信号にしたがって各種の処理を実行する。

【００３１】ところで、前述した一連の命令処理動作実行中に、実行ユニット１０１に内蔵される命令実行開始検出手段１１２、分岐方向検出手段１１１、動的分岐検出手段１１３は、以下の動作を行う。

【００３２】まず、命令実行開始検出手段１１２は、実行ユニット１０１による新たな命令の実行開始を検出する毎に、命令ステップ端子１２２に命令ステップ信号ＩＳＴＥＰを出力する。

【００３３】また、分岐方向検出手段１１１は、実行する命令が分岐命令の場合にその分岐方向（分岐の成立または不成立）を検出して分岐成立／分岐不成立を示す信号ＢＤＩＲを分岐方向表示端子１２１に出力する。

【００３４】一方、動的分岐検出手段１１３は、分岐先を静的に計算できない分岐（レジスタ間接分岐や例外処理プログラムへの分岐）の発生を検出すると、トレース・トラップ割り込み処理を起動する。トレース・トラップ割り込み処理は、通常の割り込み要求と同様に、現在のプログラム・カウンタ（ＰＣ）、およびプログラム・ステータス・ワード（ＰＳＷ）の内容を退避するとともに、あらかじめ定められたアドレスに分岐して、トレース・トラップ処理プログラムへと制御を移す。

【００３５】以下では、本実施例の構成を持つマイクロプロセッサを用いて構成したトレース・システムの動作について説明する。

【００３６】図２は、図１の構成を持つマイクロプロセッサを使用して、キャッシュ・メモリを内蔵するマイクロプロセッサのトレースを実現するシステムの構成を示すものである。

【００３７】２０１は本実施例の構成を持つマイクロプロセッサ、２０２はメモリ、２０３はトレース処理を行うトレース・アナライザ、２０４はトレース結果を格納するメモリである。マイクロプロセッサ２０１、メモリ２０２、トレース・アナライザ２０３、メモリ２０４はデータバス、アドレスバス、ステータス信号によりそれぞれ接続されている。また、マイクロプロセッサ２０１の出力する分岐方向表示信号２１１および命令ステップ信号２１２は、トレース・アナライザ２０３に入力される。

【００３８】トレース・アナライザ２０３は、マイクロプロセッサ２０１のプログラムを静的に解析する能力を持つ。メモリ２０２上に格納されているマイクロプロセッサ２０１の実行プログラムを解析して、命令処理の流れ、（フロー）を抽出し、静的な命令処理の流れを再構成することができる。

【００３９】それでは、図２に示すトレース・システムの動作について簡単に説明する。

【００４０】まず、命令コードのトレースを開始する準備として、トレース・アナライザ２０３は、メモリ２０２に格納された実行プログラムを読み出して解析し、実行プログラム中の分岐命令アドレス、分岐先のアドレス等トレースに必要となる情報を生成し、その内部に保持する。

【００４１】マイクロプロセッサ２０１が実行プログラム処理を開始すると、トレース・アナライザ２０３は、アドレスバスおよびステータスを監視して、マイクロプロセッサ２０１のリプレース・バスサイクルを観測することにより、実行プログラムのエントリ・ポイントを決定して、メモリ２０４にそのアドレスをトレース結果として書き込む。

【００４２】リプレース・バスサイクルが終了して、マイクロプロセッサ２０１が内部命令キャッシュにキャッシングした命令の実行を開始すると、マイクロプロセッサ２０１は、前述したようにその内部の実行ユニットにおける各命令の実行開始を表示する命令ステップ信号２１２を出力しながら、内部命令キャッシュにキャッシングされた命令を実行していく。

【００４３】トレース・アナライザ２０３は、命令ステップ信号２１２を検出する毎に、内部に保持している命令処理フローの解析結果を参照して命令を１ステップづつ追跡し、追跡した各命令のアドレスをメモリ２０４にトレース結果として追加する。

【００４４】以上のようにして、マイクロプロセッサ２０１の命令キャッシュにキャッシングされている命令の実行経過を含めてトレースすることができる。

【００４５】次に、内蔵命令キャッシュにキャッシングされた命令列の中に条件分岐命令がある場合について考える。条件分岐命令の場合は、たとえ分岐先があらかじめ静的に計算できる場合であっても、命令フローが分岐成立／不成立の２つのいずれに進むかが判定できないため、前述の命令ステップ信号２１２のみでは内蔵命令キャッシュにキャッシングされた命令をトレースすることは不可能である。

【００４６】しかしながら、本実施例のトレース・システムにおいては、マイクロプロセッサ２０１は分岐方向表示信号２１１を有しており、実行中の分岐命令の分岐

方向を外部に表示する手段を持っている。

【００４７】したがって、トレース・アナライザ２０３は、マイクロプロセッサ２０１が条件分岐命令実行時に出力する分岐方向表示信号２１１を参照することで動的に決定される分岐方向を検出することができるため、キャッシングされている条件分岐命令の処理を追跡することが可能となる。

【００４８】最後に、分岐先があらかじめ静的に計算できないような分岐の場合について考える。ここでは、分岐先があらかじめ静的に計算できない分岐を発生させる例として、レジスタ間接分岐命令の実行について考えてみるにとにする。

【００４９】レジスタ間接分岐命令は、命令で指定されたレジスタの値を分岐先アドレスとして分岐する命令である。分岐先アドレスとなるレジスタの値は、通常実行時に計算されるため、プログラムの静的な解析であらかじめ計算することができない。したがって、レジスタ間接分岐命令が内蔵キャッシュにキャッシングされている場合には、トレースを継続することは不可能である。

【００５０】ここでは、マイクロプロセッサ２０１が、図３に示すようなレジスタ間接分岐命令を含む実行プログラムを実行する場合を例として、本実施例の構成を持つマイクロプロセッサを利してどのようにトレースが行なわれるかを説明することにする。

【００５１】いま、内蔵命令キャッシュに図３に示す実行プログラムをすべてにキャッシングしていると仮定する。キャッシングされた実行プログラムの命令処理実行経過は、前述したようにマイクロプロセッサ２０１が出力する命令ステップ信号２１２を利用して、トレース・アナライザ２０３はキャッシングされた実行プログラムのトレースを実行している。ここで、マイクロプロセッサ２０１がレジスタ間接分命令を実行すると、マイクロプロセッサ２０１に内蔵されている、図１の１１３にあたる動的分岐検出手段がこれを検出して、マイクロプロセッサ２０１はトレース・トラップを発生させてその制御をトレース・トラップ処理プログラムに移す。レジスタ間接分岐の分岐先アドレス（すなわち指定されたレジスタの値）はこの時すでに確定しているため、マイクロプロセッサ２０１はトレース・トラップ処理プログラム中で分岐先アドレスを格納しているレジスタの値をデータバスにタンプする等して、レジスタ間接分岐の分岐先アドレスをマイクロプロセッサ２０１の外部に出力することができる。トレース・アナライザ２０３は、このトレース・トラップ処理を検出してデータバス上に出力された分岐先アドレスを読み込んで、あらたなトレースのスタート・ポイントとして以後のトレース動作を継続する。

【００５２】上述の説明ではレジスタ間接分岐命令を例として説明したが、実行時例外の発生による例外処理プログラムへの分岐のような場合にも、上述したトレース・トラップ機構を利用して同様にトレースが可能であることは明白である。

【００５３】次に本発明の別の実施例について図４および図５を参照して説明する。

【００５４】図１に示したマイクロプロセッサでは、分岐方向表示信号および命令ステップ信号を独立した専用の端子を使用して外部に出力していた。本実施例では、分岐方向表示信号および命令ステップ信号を、バス制御ユニット４０１でエンコードし、ステータス信号線を利用してステータスの一部として外部に出力するという差異がある。

【００５５】分岐方向表示信号４２１および命令ステップ信号４２２の動作は、前記第一の実施例のそれぞれと同一である。分岐方向の表示および命令のステップ情報が、専用の端子および専用の信号線を使用せずに、バスステータス信号を利用してトレース・アナライザ５０３に伝達される点をのぞいて、第５図に示したトレース・システムが第一の実施例のトレース・システムと同一に機能することは明白である。

【００５６】

【発明の効果】以上説明したように、本発明を用いることでキャッシュ・メモリを内蔵したマイクロプロセッサに対して、内蔵キャッシュを活性にした状態における効率的なトレース機構を提供することができる。また、本発明を利用したトレース機能は、トレース・トラップ機能を利用したトレース部分を除いて、マイクロプロセッサの命令実行を実時間でトレースすることができるため、キャッシュ・メモリを内蔵したマイクロプロセッサのプログラム、あるいはシステムに対して正確で効率的なデバッグを可能とする。

【図面の簡単な説明】

【図１】本発明を利用したマイクロプロセッサの第１の実施例の構成である。

【図２】第１の実施例のマイクロプロセッサを使用したトレース・システムの構成図である。

【図３】レジスタ間接分岐命令のトレース・トラップの様子を示す図である。

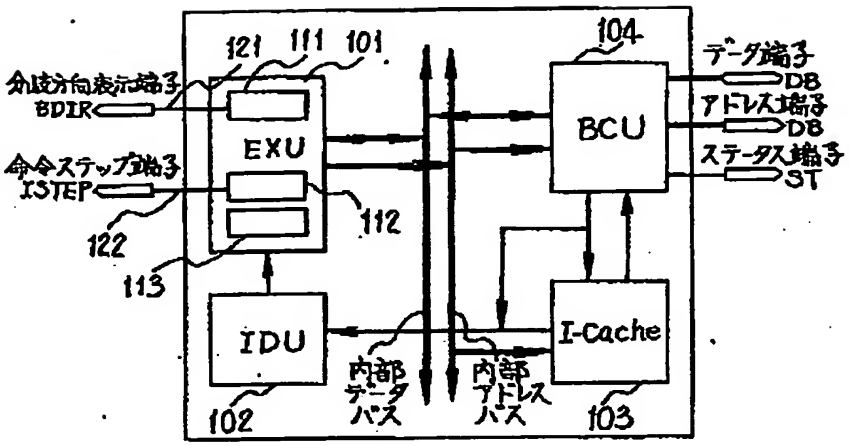
【図４】本発明を利用したマイクロプロセッサの第２の実施例の構成図である。

【図５】第２の実施例のマイクロプロセッサを使用したトレース・システムの構成図である。

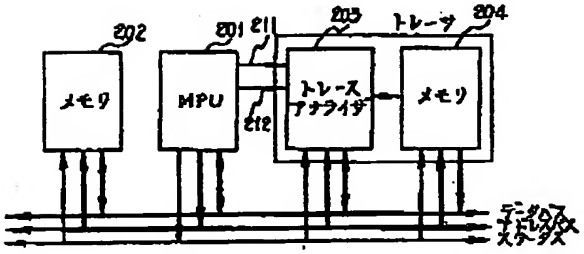
【図６】従来のキャッシュ・メモリを内蔵するマイクロプロセッサの構成図である。

【図７】従来のマイクロプロセッサを使用したトレース・システムの構成図である。

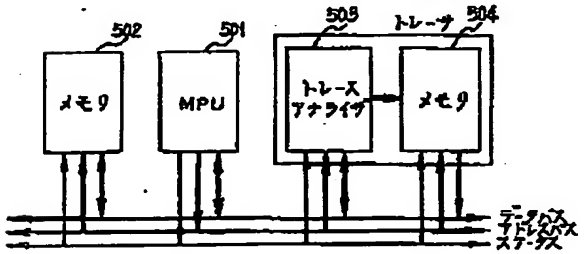
【図１】


121
111
101
104
データ端子
DB
分岐方向表示端子
BDIR
EXU
BCU
アドレス端子
DB
命令ステップ端子
ISTEP
ステータス端子
ST
112
122
113
IDU
I-Cache
内部データバス
内部アドレスバス
102
103

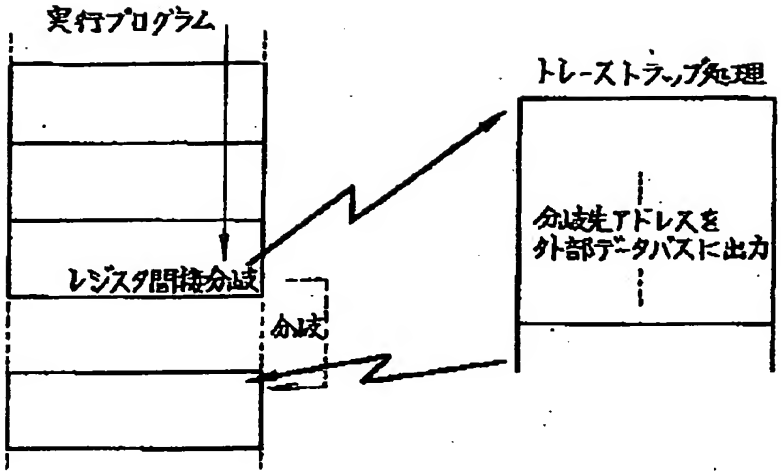
【図２】


202
201
211
205
トレーサ
204
メモリ
MPU
212
トレースアナライザ
メモリ
データバス
アドレスバス
ステータス

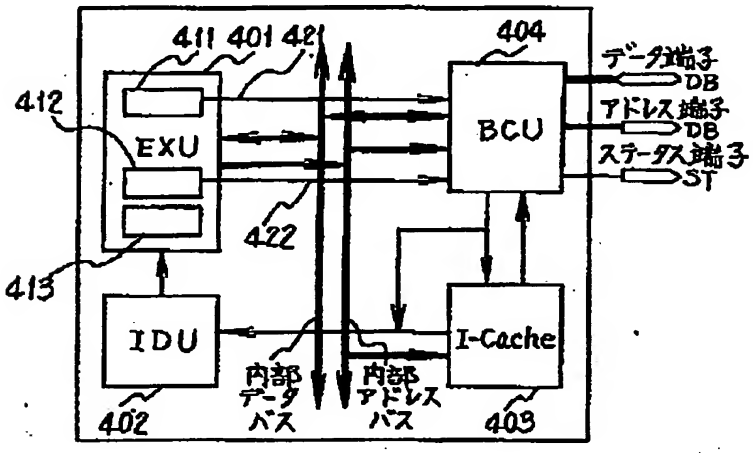
【図５】


502
501
503
トレーサ
504
メモリ
MPU
トレースアナライザ
メモリ
データバス
アドレスバス
ステータス

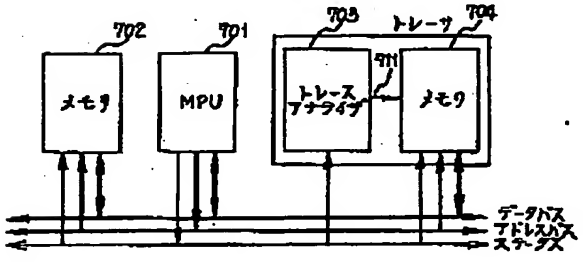
【図3】


実行プログラム
トレーストラップ処理
分岐先アドレスを
外部データバスに出力
レジスタ間接分岐
分岐

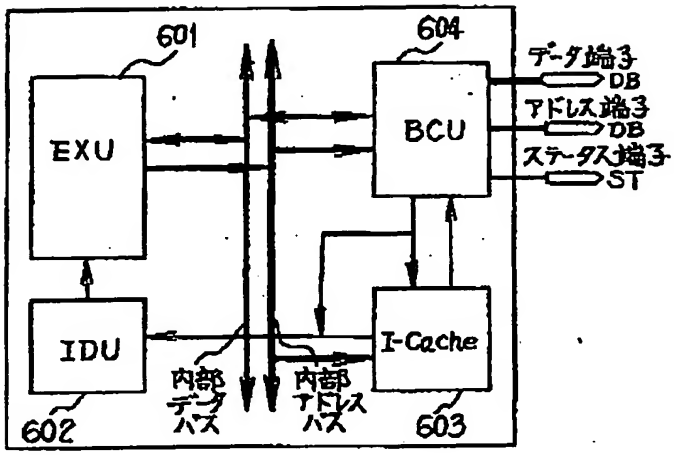
【図4】


411
401
421
404
412
EXU
BCU
データ端子
DB
アドレス端子
DB
ステータス端子
ST
422
413
IDU
I-Cache
内部
データ
バス
内部
アドレス
バス
402
403

【図7】


702
701
703
トレーサ
704
メモリ
MPU
トレース
アナライザ
711
メモリ
データバス
アドレスバス
ステータス

【図6】


601
604
EXU
BCU
データ端子
DB
アドレス端子
DB
ステータス端子
ST
IDU
I-Cache
内部データバス
内部アドレスバス
602
603

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-028002
(43)Date of publication of application : 05.02.1993

(51)Int.Cl. G06F 11/28
G06F 11/28

(21)Application number : 03-184244
(22)Date of filing : 24.07.1991

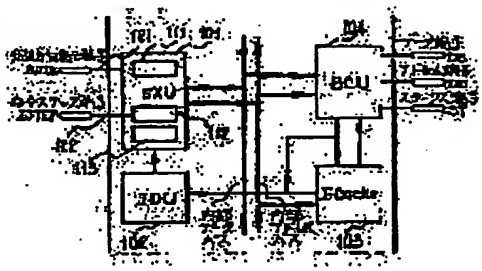
(71)Applicant : NEC CORP
(72)Inventor : ARAI TOMOHISA

(54) MICROPROCESSOR

(57)Abstract:

PURPOSE: To effectively debug a microprocessor which has a cache memory within it by providing a means for notifying an outside of starting execution of instruction, a means for notifying an outside of a branching direction of a branching instruction and a means for executing an interruption in case a program flow can not be statically calculated in advance.

CONSTITUTION: An execution unit 101 incorporates within itself a branching direction detecting means 111 which detects branching direction of a branching instruction, a starting execution of instruction detecting means 112 which detectsrespective starting of execution of instructions and a means 113 which detects occurrence of an event as to which program flow can not be statically calculated in advance. The starting execution instruction detecting means 112 outputs, each time it detects that a new instruction is executed, an instruction step signal to a terminal 122. The branching direction detecting means 111 outputs, each time it detects a branching instruction, its acknowledgement/non-acknowledgement to a terminal 121. When a dynamic branching detecting means 113 detects a branching which cannot be statically calculated, it starts a trace/ trap interruption process.



LEGAL STATUS

[Date of request for examination] 30.06.1995
[Date of sending the examiner's decision of rejection] 14.10.1997
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

CLAIMS

[Claim(s)]

[Claim 1] The microprocessor characterized by having a means to notify activation initiation of an instruction outside, a means to notify the branching direction of branch instruction outside, and a means to generate interruption when flow of a program cannot be calculated statically beforehand, in the microprocessor which built in cache memory.

[Claim 2] The microprocessor according to claim 1 characterized by using the terminal of dedication as a means to notify activation initiation of said instruction outside.

[Claim 3] The microprocessor according to claim 1 characterized by using the pulse signal outputted on the terminal of dedication as a means to notify activation initiation of said instruction outside.

[Claim 4] The microprocessor according to claim 1 characterized by using the terminal of dedication as a means to notify the branching direction of said branch instruction outside.

DETAILED DESCRIPTION

[Detailed Description of the Invention]

[0001]

[Industrial Application] This invention relates to debugging in the system development or program development of a microprocessor which built in especially cache memory about the microprocessor which built in the cache memory which carries out the cash advance of instruction code or the memory operand.

[0002]

[Description of the Prior Art] It divides roughly as the approach of debugging at the time of the program development of a microprocessor, or the system development, and there is the following.

1. Trace : collect the information which shows a program execution location, or the information about access of a memory operand, and get to know the sequence of an instruction execution or operand access by assembling this. Generally said information can consist of the address of the bus cycle which can be observed with the external terminal of a microprocessor, data, the status, etc.

[0003] Generally, since the above-mentioned information is [the exterior of a microprocessor] collectable, even if there is no function special to a microprocessor, a tracing facility is realizable. Moreover, since it haves to be generated interruption/exception and interrupted or it is not necessary to carry out queuing of the bus cycle to the program execution set as the object of debugging, there is an advantage that timing does not change in the state of a debugging condition / un-debugging.

[0004] On the other hand, it is the passive approach of only observing the condition of a program, since a specific situation is detected and it is not interrupted to the program for debugging.

2. Trap : move control to a debug program (a debugger is called below) because there was access of the instruction code of the specific location (address) set up beforehand or a memory operand, and advance still more detailed debugging.

[0005] Unlike trace, when required, control is moved to a debugger, and it has the advantage which can advance debugging. However, interruption/exception is correctly generated to the set-up address (for example, just before doing /activation of immediately after performing instruction code with the set-up address), and it is required that it should have a hardware-device (generally called a trap interrupt) for moving control in a debugger.

[0006] Moreover, if the trap as 2 schedules which needs to know the exact execution sequence of one program is not applied, it has the fault of being unable to interrupt program execution (for example, when a program overruns recklessly before a trap).

3. Single step : advance program execution, displaying the internal states (a general-purpose register, processor status word:PSW, program counter-C, etc.) of a microprocessor, or changing [move control,] some contents into a debugger, whenever it executes one instruction.

[0007] Since the internal state of a microprocessor can be grasped for every instruction execution, program execution progress can be extremely grasped in a detail. It is required that interruption/exception should be generated on the other hand when one instruction is executed to the microprocessor itself, and it should have a hardware-device (generally called single step interruption) for moving control in a debugger, and it is not the reason which can realize every microprocessor. moreover, in order that control may move to a debugger for every instruction, to the program for debugging, it is 3 interior-action timing which cannot be debugged at debugging / the time of not debugging, and 2 with 1 bad performance timing completely differs from critical processing (a timer routine etc. is mentioned as an example) -- there is also a fault of **.

[0008] Since each debugging approach has each advantage and fault as stated above, it is used combining as follows generally.

[0009] 1. Get to know in what kind of sequence the instruction is actually executed by the

tracing facility.

[0010] 2. Move control to a debugger in the neighborhood which the problem has generated by the trap function.

[0011] 3. It follows the one-instruction activation situation of an instruction at a time carefully by the single step function.

[0012] Thus, a tracing facility is a function indispensable when performing efficient program and debugging of a system.

[0013] Below, the implementation approach of a tracing facility is explained with reference to a drawing.

[0014] Drawing 6 shows the configuration of the microprocessor which built in the conventional cache memory. Moreover, drawing 7 shows the general system configuration for realizing a tracing facility.

[0015] Actuation of the microprocessor when not using a built-in cache with reference to drawing 6 and 7 first and actuation of a trace system are explained briefly. About the actuation when not using a built-in cache, it mentions later.

[0016] Drawing 6 is used first and the interior action of the microprocessor when not using a built-in cache is explained. EU 601 generates the address of the instruction executed next, and supplies it to the bus cycle control unit 604 through the internal address. The bus cycle control unit 604 outputs the address of the instruction which fetches the status signal which shows that the bus cycle to perform is an instruction fetch cycle to the status terminal ST to an address terminal AB, respectively, reads the instruction code returned to the data terminal DB, and transmits it to the instruction decoding 602. The instruction decoding 602 decodes the given instruction code, generates a control signal, and supplies it to EU 601. EU 601 performs processing according to each instruction according to the given control signal.

[0017] Then, the trace actuation in the case of tracing the instruction execution of the microprocessor mentioned above is briefly explained with reference to drawing 7 .

[0018] Since a microprocessor 701 does not use a built-in instruction cache, when it fetches an instruction, it always needs to start an instruction fetch cycle, and needs to read instruction code from memory. If an instruction fetch demand arises, a microprocessor 701 will start an instruction fetch cycle and will output a status signal and the address of an instruction to fetch, respectively. The instruction code which answers this instruction fetch cycle and memory 702 returns is read into the interior through a data bus, and the instruction corresponding to instruction code is executed. By the way, by supervising a status signal, trace ANARAZAIRA 703 on a trace system detects generating of this instruction fetch cycle, and incorporates and records the instruction address which is made to generate a strobe signal 711 and is outputted on the address bus into the instruction fetch cycle on the trace memory 704. Thus, instruction trace is created by recording all the addresses of the instruction which a microprocessor 701 executes on the trace memory 704.

[0019]

[Problem(s) to be Solved by the Invention] Now, actuation of the microprocessor in the case of returning to drawing 6 and using a built-in instruction cache is considered.

[0020] First, EU 601 generates the address of the instruction executed next, and supplies it to an instruction cache 603 through an internal address bus.

[0021] When the instruction applicable to an instruction cache 603 is registered (henceforth a "hit"), an instruction cache 603 supplies instruction code to an instruction decoder 602 immediately. It is the same as that of the actuation when not using the built-in instruction cache in which actuation was mentioned above after instruction code was supplied to an instruction decoder 602.

[0022] On the other hand, when the corresponding instruction is not registered into an instruction cache 603 (henceforth a "mistake hit"), an instruction cache 603 requires the registration to an instruction cache 603 of the bus cycle control unit 604 as the fetch from the memory of the instruction which carried out the mistake hit. The bus cycle control unit 604 starts a replacement cycle, and it supplies it to an instruction decoder 602 while outputting the address and the status to each terminal, reading the instruction code which carried out the mistake hit from a data terminal and registering with an instruction cache 603. It is the same as that of the actuation when not using the built-in instruction cache in which actuation was mentioned above after instruction code was supplied to an instruction decoder 602.

[0023] Therefore, since the information which shows advance of instruction processing is not outputted outside when the instruction cache has hit even if instruction processing inside a microprocessor advances, the advance situation of instruction processing of a microprocessor cannot be known by observing an external signal. Therefore, in the trace system of a configuration as shown in drawing 7 , it becomes impossible to trace actuation of a microprocessor.

[0024] In order to solve this problem, at the time of trace, the function of an instruction cache is made inactive, ****** of all instruction fetchs is made possible from the exterior, and how to realize a tracing facility can be considered. However, since actuation of the microprocessor at the time of trace and real actuation differs in this case, there is a fault of exact debugging becoming impossible.

[0025]

[Means for Solving the Problem] This invention has a means to notify activation initiation of an instruction outside, a means to notify the branching direction of branch instruction outside, and a means to generate interruption when flow of a program cannot be calculated statically beforehand, in the microprocessor which built in cache memory.

[0026]

[Example] Hereafter, it explains in full detail with a drawing.

[0027] Drawing 1 shows the configuration of the microprocessor using this invention. All over drawing, since it is easy, only the block relevant to this invention is shown, and it has omitted except it.

[0028] For 101, as for an instruction decoder and 103, the EU of instruction processing and 102 are [an instruction cache and 104] bus control units. These units are connected through the internal address bus and the internal data bus. EU 101 builds a branching direction detection means 111 detect the branching direction of branch instruction to the interior, an

instruction-execution initiation detection means 112 detect activation initiation of each instruction, and a means 113 (it calls with a "dynamic branching detection means" below) detect generating (for example, activation of register indirect branch instruction, branching to the exception-handling program by exceptional generating, etc.) of the event which flow of a program cannot calculate statically beforehand.

[0029] First, actuation of this microprocessor is explained.

[0030] EU 101 opts for the instruction which should be executed next, and gives the address to an instruction cache 103 through the internal address. When HITSU [an instruction cache 103], an instruction cache 103 supplies instruction code to an instruction decoder 102 immediately. An instruction decoder 102 decodes instruction code, generates a control signal, and supplies it to EU 101. EU 101 performs various kinds of processings according to the control signal supplied from the instruction decoder 102.

[0031] By the way, the instruction-execution initiation detection means 112 built in EU 101 during a series of instruction-processing actuation activation mentioned above, the branching direction detection means 111, and the dynamic branching detection means 113 perform the following actuation.

[0032] First, whenever the instruction-execution initiation detection means 112 detects activation initiation of the new instruction by EU 101, it outputs the instruction-step signal ISTEP to the instruction-step terminal 122.

[0033] Moreover, the branching direction detection means 111 outputs the signal BDIR which detects the branching direction (formation or failure of branching), and shows branching formation / branching failure to the branching direction display terminal 121, when the instruction to execute is branch instruction.

[0034] On the other hand, the dynamic branching detection means 113 will start trace trap interruption processing, if generating of branching (register indirect branching and branching to an exception-handling program) which cannot calculate a branching place statically is detected. Like the usual interrupt request, trace trap interruption processing branches to the address defined beforehand, and moves control to a trace trap processing program while it evacuates a current program counter (PC) and the contents of program status WORD (PSW).

[0035] Actuation of the trace system constituted from below using the microprocessor with the configuration of this example is explained.

[0036] Drawing 2 uses a microprocessor with the configuration of drawing 1 , and shows the structure of a system which realizes trace of the microprocessor which builds in cache memory.

[0037] The trace analyzer with which the microprocessor in which 201 has the configuration of this example, and 202 perform memory, and 203 performs trace processing, and 204 are memory which stores a trace result. A microprocessor 201, memory 202, the trace analyzer 203, and memory 204 are connected by the data bus, the address bus, and the status signal, respectively. Moreover, the branching direction status signal 211 and the instruction-step signal 212 which a microprocessor 201 outputs are inputted into the trace analyzer 203.

[0038] The trace analyzer 203 has the capacity to analyze the program of a microprocessor 201 statically. The executive program of the microprocessor 201 stored on memory 202 can be

analyzed, the flow of instruction processing and a (flow) can be extracted, and the flow of static instruction processing can be reconfigurated.

[0039] Then, actuation of the trace system shown in drawing 2 is explained briefly.

[0040] First, as preparation which starts trace of instruction code, the trace analyzer 203 reads and analyzes the executive program stored in memory 202, generates the information which is needed for traces, such as the branch instruction address in an executive program, and the address of a branching place, and holds it to the interior.

[0041] If a microprocessor 201 starts executive program processing, by supervising an address bus and the status and observing the replacement bus cycle of a microprocessor 201, the trace analyzer 203 will determine the entry point of an executive program, and will write the address in memory 204 as a trace result.

[0042] A replacement bus cycle is completed, and if activation of the instruction in which the microprocessor 201 carried out the cash advance to the internal instruction cache is started, the microprocessor 201 executes the instruction by which the cash advance was carried out to the internal instruction cache, outputting the instruction-step signal 212 which displays activation initiation of each instruction in the EU of the interior, as mentioned above.

[0043] Whenever the trace analyzer 203 detects the instruction-step signal 212, it adds the address of each instruction which pursued one step of instructions at a time, and pursued them with reference to the analysis result of the instruction-processing flow currently held inside to memory 204 as a trace result.

[0044] It is traceable including activation progress of the instruction by which the cash advance is carried out as mentioned above to the instruction cache of a microprocessor 201.

[0045] Next, the case where a conditional-branching instruction is in the instruction train by which the cash advance was carried out to the built-in instruction cache is considered. In a conditional-branching instruction, since to branching formation / two abortive any an instruction flow progresses cannot judge even if it is the case where a branching place can calculate statically beforehand even if, it is impossible for tracing the instruction by which the cash advance was carried out to the built-in instruction cache only by the above-mentioned instruction-step signal 212.

[0046] However, in the trace system of this example, the microprocessor 201 has the branching direction status signal 211, and has a means to display the branching direction of the branch instruction under activation outside.

[0047] Therefore, since a microprocessor 201 can detect the branching direction determined dynamically by referring to the branching direction status signal 211 outputted at the time of a conditional-branching instruction execution, the trace analyzer 203 becomes possible [pursuing processing of the conditional-branching instruction by which the cash advance is carried out].

[0048] Finally, the case where it is branching which a branching place cannot calculate statically beforehand is considered. as the example which generates branching which a branching place cannot calculate statically beforehand here ·· activation of register indirect branch instruction -- thinking -- ** -- it is alike and carries out.

[0049] A register indirect decision instruction is an instruction which branches considering the value of the register specified with an instruction as the branching place address. Since the value of the register used as the branching place address is usually calculated at the time of activation, it is incalculable beforehand in the static analysis of a program. Therefore, it is impossible to continue trace, when the cash advance of the register indirect branch instruction is carried out to the built-in cache.

[0050] It will explain how here, a microprocessor with the configuration of this example is benefited by making into an example the case where the executive program in which a microprocessor 201 includes register indirect branch instruction as shown in drawing 3 is performed, and trace is performed.

[0051] It is assumed that the cash advance of the executive program shown in a built-in instruction cache at drawing 8 is now carried out to all. The trace analyzer 203 is performing trace of an executive program by which the cash advance was carried out using the instruction-step signal 212 which a microprocessor 201 outputs as the instruction-processing activation progress of an executive program by which the cash advance was carried out was mentioned above. Here, if a microprocessor 201 executes a register indirect part instruction, a dynamic branching detection means built in the microprocessor 201 to hit 113 of drawing 1 detects this, and a microprocessor 201 will generate a trace trap and will move the control to a trace trap processing program. Since it has already decided at this time, a microprocessor 201 can carry out TAMPU [the value of the register which stores the branching place address in a trace trap processing program / a data bus] etc., and the branching place address (namely, value of the specified register) of register indirect branching can output the branching place address of register indirect branching to the exterior of a microprocessor 201. The trace analyzer 203 reads the branching place address which detected this trace trap processing and was outputted on the data bus, and continues future trace actuation as the starting point of new trace.

[0052] Although above-mentioned explanation explained register indirect branch instruction as an example, also case [like branching to the exception-handling program by generating of an execution-time exception], it is clear for it to be able to trace similarly using the trace trap device mentioned above.

[0053] Next, another example of this invention is explained with reference to drawing 4 and drawing 5 .

[0054] In the microprocessor shown in drawing 1 , the branching direction status signal and the instruction-step signal were outputted outside using the terminal of the independent dedication. In this example, the branching direction status signal and an instruction-step signal are encoded in the bus control unit 404, and there is a difference of outputting outside as a part of status using a status signal line.

[0055] Actuation of the branching direction status signal 421 and the instruction-step signal 422 is the same as that of each of said first example. It is clear that remove the point that the display of the branching direction and the step information on an instruction are transmitted to the trace analyzer 503 using a bus status signal, without using the terminal of dedication

and the signal line of dedication, and the trace system shown in Fig. 5 functions identically to the trace system of the first example.

[0056]

[Effect of the Invention] As explained above, the efficient trace device in the condition of having made the built-in cache into activity to the microprocessor which built in cache memory by using this invention can be offered. Moreover, since the tracing facility using this invention can trace the instruction execution of a microprocessor in the real time except for the trace part using a trace trap function, it enables exact and efficient debugging to the program of the microprocessor which built in cache memory, or a system.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]

[Drawing 1] It is the configuration of the 1st example of the microprocessor using this invention.

[Drawing 2] It is the trace structure-of-a-system Fig. which used the microprocessor of the 1st example.

[Drawing 3] It is drawing showing the situation of the trace trap of register indirect branch instruction.

[Drawing 4] It is the block diagram of the 2nd example of the microprocessor using this invention.

[Drawing 5] It is the trace structure-of-a-system Fig. which used the microprocessor of the 2nd example.

[Drawing 6] It is the block diagram of the microprocessor which builds in the conventional cache memory.

[Drawing 7] It is the trace structure-of-a-system Fig. which used the conventional microprocessor.

[Translation done.]